Demo Abstract: *Stemflow:* Inter-Datacenter Overlay as a Service

Shuhao Liu, Baochun Li
Department of Electrical and Computer Engineering, University of Toronto
{*shuhao, bli*}@ece.toronto.edu

***Abstract*—Thanks to the abundant available bandwidth and multiple paths on wide-area links that interconnect datacenters on major cloud platforms, it is conceivable that bandwidth-intensive applications may improve their performance by relaying their traffic through such an inter-datacenter network. We propose *Stemflow*, a new systems framework that provides *Inter-Datacenter Overlay as a Service*. It is provided as a turn-key solution for applications to conveniently and transparently tap into the available bandwidth resources across datacenters in the cloud, with internal optimizations on the performance of data delivery. Internally, we have implemented Stemflow as a software-defined overlay. We present two interactive cloud applications built upon Stemflow — live video broadcast and instant messaging — to demonstrate the ease of development and the performance gains by using Stemflow.**

I. Introduction

Cloud service providers, such as Amazon and Google, have routinely provided convenient access to both computation and network resources in their datacenters with affordable costs. These datacenters are geographically distributed across the world, and are often inter-connected with high-capacity links. Some inter-datacenter links have the ability to provide 100s of Gbps to Tbps of capacity, due to the use of dedicated fiber-optic links [1], [2]. For this reason, such *inter-datacenter networks* have recently emerged as an attractive option to serve as the "backbone" of the public Internet over the wide area.

In this paper, we present our design of *Stemflow*, a new systems framework that provides an *Inter-Datacenter Overlay as a Service*. Much in the spirit of Software-as-a-Service, it provides a cloud service that offers a simple turn-key solution for an end application to improve its scalability and performance by using the inter-datacenter network for its traffic, yet with very simple and concise interfaces.

Internally, *Stemflow* is implemented as a *software-defined* [3] overlay atop an inter-datacenter network. The overlay consists of a number of virtual machines located at geographically distributed datacenters, as traffic relays. Among these relays, persistent, parallel TCP connections are established for maximized throughput, which are multiplexed among applications. Moreover, with a complete separation of the control and data planes, the routing decisions to be used for relaying application traffic are fully controlled by a centralized controller. For the sake of convenience, all of these design choices and their implementation details are completely hidden from the perspective of application developers.

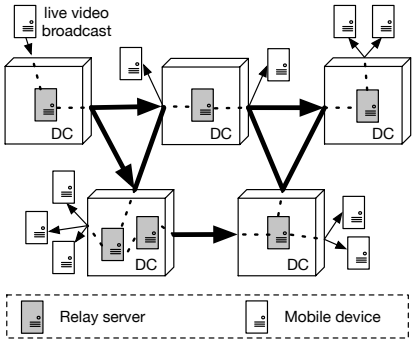

Fig. 1: An example deployment of a video broadcast application across five datacenters. Video streams are multicast through high-capacity inter-datacenter links to achieve better performance.

Although *Stemflow* is simple from the perspective of applications, it is designed to achieve high performance and to scale well, offering an attractive alternative to applications in need for performance. Enjoying the flexibility of an application-layer software-defined overlay, *Stemflow* is designed to provide native support for multicast sessions with multiple paths and trees.

To demonstrate the simplicity and performance gains of using *Stemflow* as an Inter-Datacenter Overlay as a Service, we have developed two example bandwidth-intensive and interactive applications, video broadcast and instant messaging, as simple web-based clients. As shown in Fig. 1, generated network traffic, including image and video file transfers and live video streams, is completely handled by *Stemflow* across multiple datacenters. Both applications demonstrate a high level of Quality-of-Experience, with minimal perceived latencies and high throughput in multicast data delivery across users that are geographically distributed.

II. Motivation

There exist mobile applications — most prominently live video broadcast and high-quality video conferencing — that are *bandwidth-intensive*. They typically depend on application-specific solutions that are based on centralized servers, and video streams are relayed by one of the servers from the video source to the participants of the session. Referred to as server-based solutions, existing solutions heavily depend on the accuracy of selecting the best possible server to assist a live video broadcast or conferencing session. Unfortunately, for sessions involving a large number of participants that are geographically distributed across the world, it may happen that

Solution	Throughput (Mbps)	Latency (ms)
Direct TCP	181	395.9
Stemflow	516	468.5
	+285.1%	**+72.6 ms**

TABLE I: A comparison of the average throughput and latencies, between US-East and Asia-East regions on the Google Cloud platform, between using a direct TCP connection and Stemflow. The average throughput is evaluated by sending 1 GB of data, while the latency is measured by sending 1 KB of data between end hosts.

a single server may not offer satisfactory performance due to the lack of available bandwidth to some of the participants.

Stemflow is designed as an application framework to assist these bandwidth-intensive applications, offering them better performance by using the inter-datacenter network. Intuitively, as compared to traditional server-based solutions, there are two reasons why an inter-datacenter network may be able to offer better performance. First, at a per-GB cost of around ten cents, inter-datacenter links often offer much higher bandwidth capacities (with multiple inter-datacenter paths available), up to hundreds of Mbps. Second, rather than selecting a server for all the participants to connect to, each participant is able to connect to a datacenter that is considered its best choice in terms of bandwidth or latency. From a workload perspective, this is naturally a more scalable solution.

III. Features

High end-to-end throughput. Stemflow focuses on exploiting the full potential of the inter-datacenter network. Persistent, parallel TCP connections are used to provide the best possible throughput in such a high bandwidth-delay product (BDP) environment. Moreover, our software-defined architecture supports a high degree of flexibility when making routing decisions, such as the use of multiple paths across datacenters. Table I shows that Stemflow significantly improves the throughput of data transfers across continents.

Minimal latency overhead. When implementing Stemflow, we have designed its architecture and optimized its implementation with a focus on its performance and overhead. We used code-level optimization techniques such as zero-copying and asynchronous network I/O, and the entire implementation is completed in C++.

Simple programming interfaces that are easy to use. Optimized performance and scalability can only become attractive if application developers find Stemflow simple to use as a framework. From the perspective of the developers, it would be ideal if Stemflow can be viewed as a single server, with Publish/Subscribe-style APIs.

IV. Internal Architectural Design

Stemflow is designed internally as a software-defined application-layer overlay: it employs relay servers in the data plane to interact with applications and to forward traffic, and includes a central controller in the control plane to make routing and scheduling decisions.

Relay servers. Stemflow is deployed within Virtual Machines (VMs) or Docker containers, which are initiated across geo-distributed centers. There may be multiple VMs or containers located at each datacenter. These VMs or containers are typically leased from a public cloud provider, such as Google

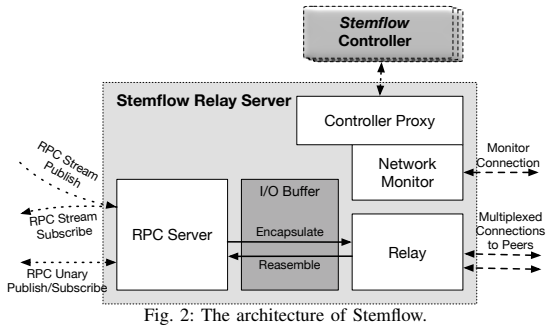

Fig. 2: The architecture of Stemflow.

Cloud or Amazon EC2. The number of VMs or containers used depends on the workload at runtime, and can be adjusted dynamically over time. Within each VM or container, the *Stemflow* runtime executable is simply called a *relay server*. As shown in Fig. 2, the relay servers work as application-layer switches, with close interactions with the Controller. It has a RPC-based Pub/Sub interface that supports both streaming and unary data transfers, while maintaining multiple persistent, parallel TCP connections to other peer relay servers.

Central controller. With software-defined networking as its underlying principle, the control and data planes in Stemflow are fully decoupled. The control plane is implemented with a centralized controller in Javascript and Python, and it communicates proactively with all the relay servers, collecting real-time measurement statistics and deploying its control decisions.

Based on the collected measurements such as real-time available bandwidth and latencies, the controller is able to make dynamic routing and scheduling decisions, with the objective of optimizing the Quality-of-Experience of the applications. It is easy to enable multi-path routing among datacenters to offer maximized throughput, or to prioritize the traffic for delay-sensitive applications.

V. Highlights of Stemflow Applications

We present two web-based applications using the Stemflow framework, video conferencing and instant messaging. Both applications require multicast and are bandwidth-intensive, while video conferencing is more delay-sensitive. We show that both applications perform well with the help of Stemflow, with minimal perceived delay and no visual disruptions. Also, to build these applications based on an existing code base, it requires minimal changes to the source code, demonstrating the convenience of using Stemflow as a development framework.

References

[1] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven wan. In *Proc. of ACM SIGCOMM*, 2013.

[2] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al. B4: Experience with a globally-deployed software defined wan. In *Proc. of ACM SIGCOMM*, 2013.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM CCR*, 38(2):69–74, 2008.